Efficient and robust reconstruction of botanical branching structure from laser scanned points

Dong-Ming Yan
Dept. of CS
Univ. of Hong Kong
Hong Kong, China
dmyan@cs.hku.hk

Julien Wintz, Bernard Mourrain
Project Galaad
Inria
Sophia-Antipolis, France
mourrain,wintz@sophia.inria.fr

Wenping Wang
Dept. of CS
Univ. of HongKong
Hong Kong, China
wenping@cs.hku.hk

Frédéric Boudon and Christophe Godin
Project Virtual Plants
Inria
Montpellier, France
boudon,godin@inria.fr

Abstract

This paper presents a reconstruction pipeline for recovering branching structure of trees from laser scanned data points. The process is made up of two main blocks: segmentation and reconstruction. Based on a variational k-means clustering algorithm, cylindrical components and ramified regions of data points are identified and located. An adjacency graph is then built from neighborhood information of components. Simple heuristics allow us to extract a skeleton structure and identify branches from the graph. Finally, a B-spline model is computed to give a compact and accurate reconstruction of the branching system.

1. Introduction

Due to the complexity and the diversity of plant shapes, the construction of plant geometric models is still a challenging problem for both computer graphics and biology. In this paper, we consider the problem of reconstructing complete faithful branching systems of observed tree from 3D laser scans. Since branches may be hidden by leaves, we restrict our approach to trees without leaves (e.g. temperate species observed in winter). The main contributions of this paper include:

- A complete framework for branch structure reconstruction from scanned point cloud;
- A new variational point cloud segmentation framework is proposed (Section 2), which locates the cylindrical and branching regions efficiently;
- Simple heuristics are proposed to reconstruct branching structure of trees from segmentation result (Section 3);

1.1. Related work

This work is related to the geometric modeling of tree and readers may refer to [1] for a detailed survey on this topic. More precisely, we aim to reconstruct tree models from scanned point data.

A first work on this topic was made by Pyysalo *et al.* [6] that propose to reconstruct tree crowns from scanned data for canopy feature extraction. They focus on the computation of statistical information on forests and do not consider reconstruction of branching structure. Gorte and Pfeifer [3] use a 3D morphology method to segment and extract the skeleton from point data. As reported by the authors, the space and time become the bottleneck of their method, whose complexity increases with the third power of the resolution. Based on the segmentation method of [3], Pfeifer *et al.* [5] present a method to reconstruct tree models by fitting data segments with cylinders. Due to the incompleteness of the scanned data, the fitted cylinder can be far from the real branch. The most related work to ours is the one of Xu *et al.* [7]. In this work, the input scanned data are first connected together to form a neighboring graph. The main skeleton of the tree is produced by clustering points with a given quantized distance to the root on a neighboring graph. The clusters are used to generate the tree skeleton. The resulting skeleton is however not always geometrically consistent and may contains loop because of their simple clustering procedure.

1.2. Outline

Our approach is composed of two main steps: segmentation and branch reconstruction. The overall reconstruction process is illustrated on an apple tree in Fig.1. Section 2 introduces a new variational approach for point data segmentation. Section 3 presents a procedure for reconstructing branches from the clusters obtained in the segmentation phase. Experimental results are given in Section 4 before we draw our conclusions in Section 5.

2. Segmentation

Laser scanned real trees produce a huge amount of unstructured data, which naturally feature many cylindrical shapes – the branches. The aim of the segmentation step is

978-1-4244-3701-6/09/$25.00 ©2009 IEEE

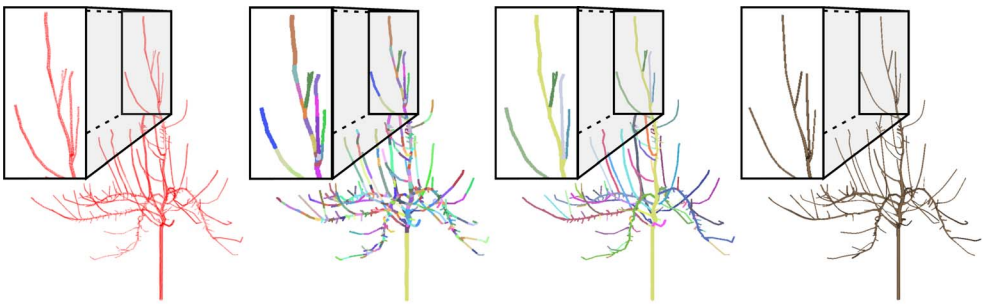


Figure 1: Tree model reconstruction. From left to right: 1) Scanned data of an apple tree; 2) Segmentation result, different colors means different clusters; 3) Branch identification result. The color of each branch is the same as its starting cluster; 4) Final B-spline surface representation of tree branches.

to partition the data into connected sub clusters, each cluster being bounded by a circular cylinder as tightly as possible.

Let $\mathcal{P} = \{p_i,\ i = 1, \ldots, N\}$ be a dense point set. A segmentation $\mathcal{C}$ of $\mathcal{P}$ is a set of clusters $\{\mathcal{C}_i,\ i = 1, \ldots, n\}$, such that $\bigcup_{i=1}^{n} \mathcal{C}_i = \mathcal{P}$, where $(n \ll N)$, and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, for any $i \neq j$. Each cluster $\mathcal{C}_i$ contains a sub set of the point set, *i.e.* $\mathcal{C}_i = \{p_k^i,\ k = 1, \ldots, n_i\}$, where $n_i = |C_i|$. We propose a hybrid approach for segmenting the point set and detecting cylinder components simultaneously. The following are the main steps of our segmentation algorithm:

1) **Preprocessing**: A kd-tree representation of the point set $\mathcal{P}$ is constructed to facilitate the retrieval of neighboring points of any point of $\mathcal{P}$.

2) **Variational clustering**: The preprocessed point set is segmented into clusters using the flooding algorithm proposed in [2]. Each cluster is a connected sub-graph of the kd-tree.

3) **Cylinder detection**: For each cluster $\mathcal{C}_i$ obtained from the variational clustering, a minimal bounding cylinder is computed to measure the tightness of the bounding volume to the point set. If some distance criteria of the point set to the bounding volume are satisfied, the cluster $\mathcal{C}_i$ is flagged as fixed and will not change anymore.

4) **Subdivision**: If some clusters cannot be bounded tightly by a cylinder or their number of points is above a given threshold, they are subdivided and the process goes back to step 2. Otherwise, the algorithm terminates.

2.1. Preprocessing

Since a point set features no adjacency relationship, the neighborhood of a point p_i is defined by its n-nearest neighbors $N_n(p_i)$. A kd-tree data structure is constructed for efficient neighborhood search. We choose neighbor size $n = 8$ in all our experiments.

2.2. Variational clustering

In this step, we want to segment the input point cloud $\mathcal{P}$ into a set of clusters $\mathcal{C}$, reflecting the main structure of the tree, where all points belonging to a cluster form a connected component. The energy function of a segmentation using k-means clustering is defined such as:

$$E(\mathcal{P}, \mathcal{C}) = \sum_{i=1}^{n} E(\mathcal{P}_i, \mathcal{C}_i) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} d(p_j, c_i)^2, \qquad (1)$$

where c_i is the center of the cluster $\mathcal{C}_i$ and $d(p_j, c_i)$ is the Euclidean distance between the point p_j and the center c_i.

To this end, we adapt the distortion minimization flooding algorithm used in [2] to efficiently segments the point set while keeping the connectivity of each cluster as well. The Lloyd iteration [4] is used to iteratively cluster points and update seeds. Here is a detailed description of our variational point set segmentation algorithm.

1) **Initialization**: To start the algorithm, we randomly select n points $\{s_i, i = 1...n\}$ as initial seeds to define n clusters centroid $\{\mathcal{C}_i, i = 1...n\}$, where n is an user input parameter.

2) **Clustering**: We segment the point set into clusters according to the seeds. For this, we associate each point with the cluster with closest seed. The implementation of the clustering algorithm is similar to that proposed in [2]. We use kd-tree instead of the connectivity information of triangle mesh. See [2] for more details.

3) **Seed update**: Once the clusters $\mathcal{C}_i$ have been identified, seed points s_i can be updated to be close to centroid of $\mathcal{C}_i$. For this, we compute the centroid c_i of each clusters $\mathcal{C}_i$ and assign s_i to the point $p_j \in \mathcal{C}_i$ being the closest point of c_i i.e. $s_i = \underset{p_j \in C_i}{\arg\min}\ d(c_i, p_j)$.

4) **Energy evaluation**: The total energy $E_t(\mathcal{P}, \mathcal{C})$ of the current clustering is then evaluated. t represents the number of iterations done so far. If a maximal number of iterations is reached or the error between two iterations is smaller than a user defined threshold (*i.e.* $|E_t - E_{t-1}| < \epsilon$), the iteration stops; otherwise process go to step 2.

2.3. Cylinder detection

The k-means clustering terminates when it has produced a well shaped partition of the input point cloud. Since the shape of branches is naturally cylindrical (at least locally), we extract cylinder components from the partition.

In order to compute bounding cylinders, the principal direction of each cluster $\mathcal{C}_i$ is computed. The eigenvector of the covariance matrix of $\mathcal{C}_i$ with largest eigenvalue is selected as the axis of the cylinder. Each point of $\mathcal{C}_i$ is projected onto the plane passing through center of the cluster $\mathcal{C}_i$ and having the axis as its normal direction. A minimal bounding circle of projected 2D points is computed, and the radius of this circle is used as the radius of the bounding cylinder. The tightness of the bounding cylinder is measured by evaluating the root mean square (RMS) error between the data points and the bounding cylinder of $\mathcal{C}_i$. If the RMS error is smaller than a threshold δ, the cluster $\mathcal{C}_i$ is flagged as fixed and will not be changed any further.

The segmentation terminates whenever all the clusters are flagged as fixed. But this case seldom happens in practice, since clusters including intersection regions of different branches are hardly bounded by a cylinder appropriately.

2.4. Cluster subdivision

In this step, each unfixed cluster is subdivided into two clusters if it has a minimal number of points (specified by a user value). The first sub-cluster is assigned the seed of its parent as its own seed. The seed of the other sub-cluster is chosen as the point in the cluster with maximal distance to the first seed. After subdivision, the process starts back at the clustering step (section 2.2). The algorithm terminates when no cluster can be subdivided anymore or the total number of clusters exceeds a maximum user specified value.

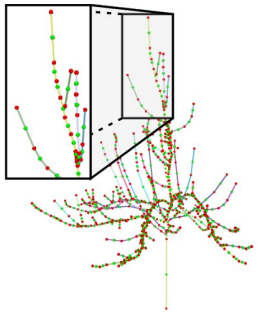
If required, user may control the clustering by inserting new cluster, merging pair of clusters of stopping locally the subdivision.

3. Branch reconstruction

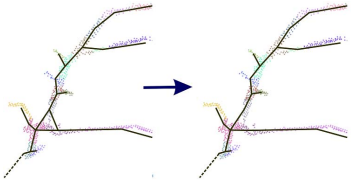
After the segmentation step, most of cylindrical regions of the input data are identified, and branching points are also located progressively. In this step we first extract the skeleton of the tree from segmented clusters. Then we identify all the branches from the skeleton and finally compute a B-spline model for the branching system.

3.1. Skeleton extraction

We construct the tree skeleton from the adjacency graph G between clusters. The adjacency graph G is built by determining the neighborhood information at point scale for each cluster. Each point p_i is compared with its n-nearest neighbors $p_j \in N_n(p_i)$: if p_i and p_j belong to different clusters $\mathcal{C}_i$ and $\mathcal{C}_j$, then $\mathcal{C}_i$ and $\mathcal{C}_j$ are flagged as neighbors. The adjacency graph G is defined with a node for each cluster and an edge for each pair of neighboring clusters.



In order to capture accurately branch curvature, we augment G with *junction points* between pairs of neighboring clusters (red points in right figure). We define junction points as centers of all boundary points *i.e.* points from two connected clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ which has at least one neighbor into the other cluster. In addition to junction points, we also add an extra point at each leaf node to extend the skeleton toward the tree extremities.



Due to the definition of neighboring cluster, the adjacency graph may not be a tree graph, since loops may appear, as shown in the left figure of inset. We resolve this problem by computing a minimum spanning tree from the adjacency graph (right figure of the inset). For this, the edges of the graph are weighted with euclidean distance between nodes.

3.2. Branch identification

A branch is defined as a sequence of nodes, starting from a branching node (*i.e.* of valence $val(N_i) \geq 3$) or root node and ending at a leaf node.

To identify all the branches, we compute recursively a hierarchy of longest paths in the tree graph. Our procedure start by determining the longest possible path starting from the root. This path will be considered as a main branch. We then remove the edges of the corresponding path on the tree graph and consider then all the branching points of the created branch as new roots for lateral branches. For each root point, we compute the longest path on the remaining forest of nodes. We repeat this procedure until all edges are assigned to a branch. It is implemented using a FIFO queue to store and examine all root points of branches.

3.3. B-spline lofting

The final step of our pipeline aims at providing a compact and smooth surface representation of trees. We first smooth the skeleton by fitting to each branch a B-spline curve of degree 3. We then compute a B-spline lofting surface by sweeping a circle along the curve of each branch. The radii of a branch along its central path is obtained by linearly interpolating the radii of the bounding cylinders along the skeleton.

4. Results and discussion

This work has been implemented as a plugin in the algebraic geometric modeling environment AXEL[1], which allows the visualization and manipulation of geometric objects with algebraic representation such as implicit or parametric curves or surfaces.

Figure 2: Reconstruction result of a digitized melgueil tree. Branch detection result (left), skeleton (middle) and the lofting result (right).

Figure 3: Reconstruction result of a digitized walnut tree. Branch detection result (left), skeleton (middle) and the lofting result (right).

The input point cloud is normalized into the unit cube $[0,1]^3$ at the initialization stage. The threshold ϵ (Section 2.2) for clustering is set to $1e^{-5}$ in all our experiments. The error threshold δ (Section 2.3) is different for models with different noise levels and varies in the interval $[1e^{-3}, 1e^{-2}]$. Several results of complex models are given in Fig.1, Fig.2 and Fig.3. Table 1 lists the timing statistic of our algorithm tested on those models. We can see that our method could process point sets with large size in a reasonable time and produce accurate results.

Model	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{B}\rvert$	T_s	T_b	T_m
apple tree	60126	356	96	26.5	0.048	6.6
melgueil tree	117601	370	103	73.4	0.078	9.255
walnut tree	217187	1394	502	215.2	1.034	114.4

Table 1: Timing Statistics (in seconds). $\lvert\mathcal{P}\rvert$ is the number of points obtained from the scan, $\lvert\mathcal{C}\rvert$ is the number of computed clusters and $\lvert\mathcal{B}\rvert$ is the number of branches. T_s, T_b and T_m are the time taken by clustering, branch reconstruction and B-spline lofting, respectively.

A comparison of skeletons generated with simple distance quantization [7] and our method based on variational clustering is given in Fig.4. Using the same number of clusters, our method yields to more faithful representations of reconstructed skeleton.

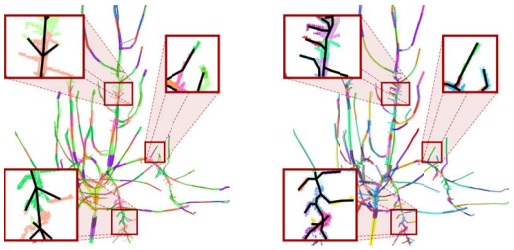


Figure 4: Comparison with previous works. The background represent the clustered point data overlaid with the skeleton. Left: Skeleton generated using method of [7] with 357 clusters. Right: skeleton using our method with 356 clusters.

5. Summary and outlook

In this paper, we present an efficient framework for reconstructing ramified branch structure from scanned point cloud data. The whole process is automatic but can possibly be dynamically controlled by the user easily. In the future, we plan to improve our reconstruction pipeline to be more robust to holes in the dataset due to occlusion during scanning. We also plan to process scanned point data with leaves.

Acknowledgments

This paper is dedicated to the memory of our friend and colleague Hervé Sinoquet who inspired this work.

References

[1] F. Boudon, A. Meyer, and C. Godin. Survey on Computer Representations of Trees for Realistic and Efficient Rendering. Technical Report RR-LIRIS-2006-003, 2006.

[2] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, 2004.

[3] B. Gorte and N. Pfeifer. Structuring laser-scanned trees using 3D mathematical morphology. In *Proc. of 20th ISPRS Congress*, pages 929–933, 2004.

[4] S.P. Lloyd. Least square quantization in PCM. *IEEE Trans. Inform Theory*, 28:129–137, 1982.

[5] N. Pfeifer, B. Gorte, and D. Winterhalder. Automatic reconstruction of single trees from terrestrial laser scanner data. In *In Proceedings of 20th ISPRS Congress*, pages 114–119, 2004.

[6] U. Pyysalo and H. Hyypp. Reconstructing tree crowns from laser scanner data for feature extraction. In *Proceedings of ISPRS Commission III*, pages 218–221, 2002.

[7] H. Xu, N. Gossett, and B. Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics*, 26(4):303–308, 2007.

1. http://axel.inria.fr